Virtual Prototyping (ViPro) Tool for Memory Subsystem Design Exploration and Optimization

Project Report: December 31, 2012

PI:
Benton Calhoun University of Virginia bcalhoun@virginia.edu (434) 243-2076

Industrial Liaisons:
Shih-lien Lu (Intel)

Task ID:
2233.001

Deliverables Met, due 8/31/12:
Deliverable Y1.1. Report describing memory array optimization across multiple technologies using an optimization engine (e.g. MOSEK)

Executive Summary
The primary goal of this research is to develop a tool for fast, efficient optimization and generation of memory macros and subsystems targeting diverse system level requirements across multiple process technology generations. Specifically, the tool will enhance the ability of a knowledgeable human designer to generate base case memories in a new technology, estimate macro level metrics before sub-components are completed or designed, compute the effects of a low level circuit change on the overall macro, and rapidly evaluate varying prototypes of arrays using new circuits or new cell technologies. In short, this tool will improve and optimize the memory design process across a diverse market space.

In our previous work [1], we developed the first version of the ViPro tool, which works in two phases. The first phase, called TASE [2] (Technology Agnostic Simulation Environment), combines process information with templates for common simulations to create parameterized characterizations of memory components in any given process technology. The second phase uses a hierarchical model of the memory array to optimize the design for a given set of constraints. In our last report, we have expanded the functionality of ViPro to support multi-bank architectures as well as the 8T bitcell [3]. We have modeled the components necessary (e.g. bank decoders, bank output muxes, and global interconnect) to accurately compute the global figures of merit for multi-bank SRAM macros.

In this work, we will expand upon the previous functionality of ViPro, by adding an optimization engine to more quickly generate the optimal design. We have added a simulating annealing algorithm, which will be discussed in the next section, which reduces the number of simulations required to determine the optimal configuration. As the number of design knobs increases, brute force methods because too time consuming because they require that each possible permutation be simulated. We will show in this report that using the simulated annealing algorithm greatly reduces the number of simulations required. In addition, we will introduce two new knobs to the design space and show how the optimization engine efficiently identifies the optimal design.

Optimization Framework

In this section we will discuss ViPro optimization framework. Figure 1 shows the framework. The inputs to ViPro are the optimization constraints on energy or delay, the design knobs (different SRAM components, assist features, etc.) and the input specification. ViPro supports two optimization flows, a simulation flow utilizing a characterization engine (CE) for accurate

simulation of SRAM across technologies, and a model flow using a hierarchical meta compiler (HMC) for fast evaluation of SRAM components. The Adaptive simulated annealing is the optimization engine used in the simulation flow which will be discussed in this report.

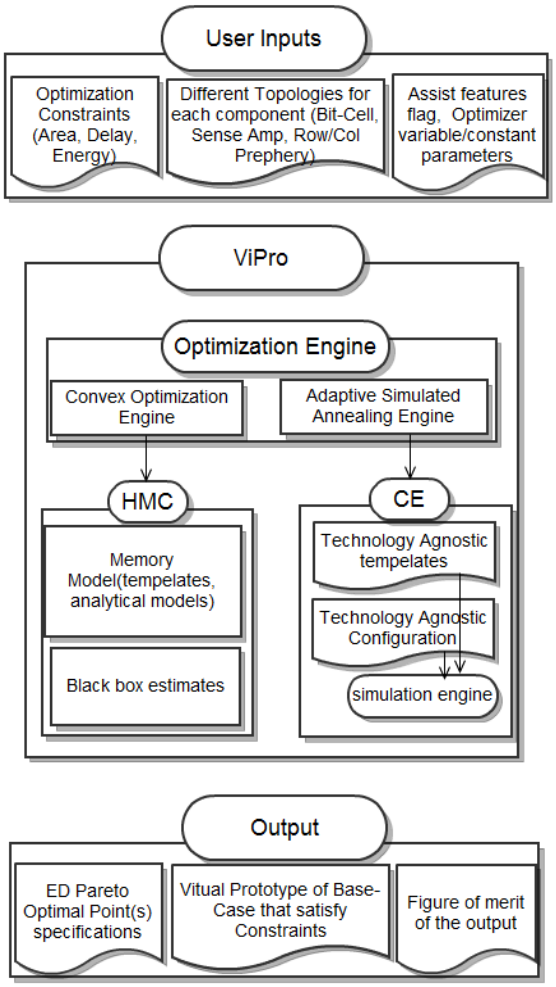


Figure 1. ViPro Framework

Figure 2 shows the flow chart of the simulated annealing algorithm used for optimization. We utilize it for SRAM optimization because of its high efficiency in optimizing large scale problems. The algorithm is an analogy of the annealing process. The "temperature" represents the probability of accepting a change. In high temperature, the elements move freely and as slow cooling, or annealing, takes place the system crystallizes into a state of minimal energy.

The algorithm starts by calculating the initial temperature, by evaluating the objective function (energy/delay) for N random SRAM designs. N is in the range of 5-10, depending on the size of the design space. This step is used to get an estimate of the range of energy/delay across the design space that the algorithm will need later.

The algorithm randomly initializes an SRAM component and evaluates the objective function (energy/delay) of the design. This step is repeated and if the new SRAM design has a better objective function, the algorithm records its data and uses it as a base design in the subsequent iteration. If the degradation in the objective function is not significant, the algorithm might

accept the design to avoid becoming locked in a local minimum point. This decision is based on the initial estimate of the range of energy/delay calculated at the beginning of the algorithm.

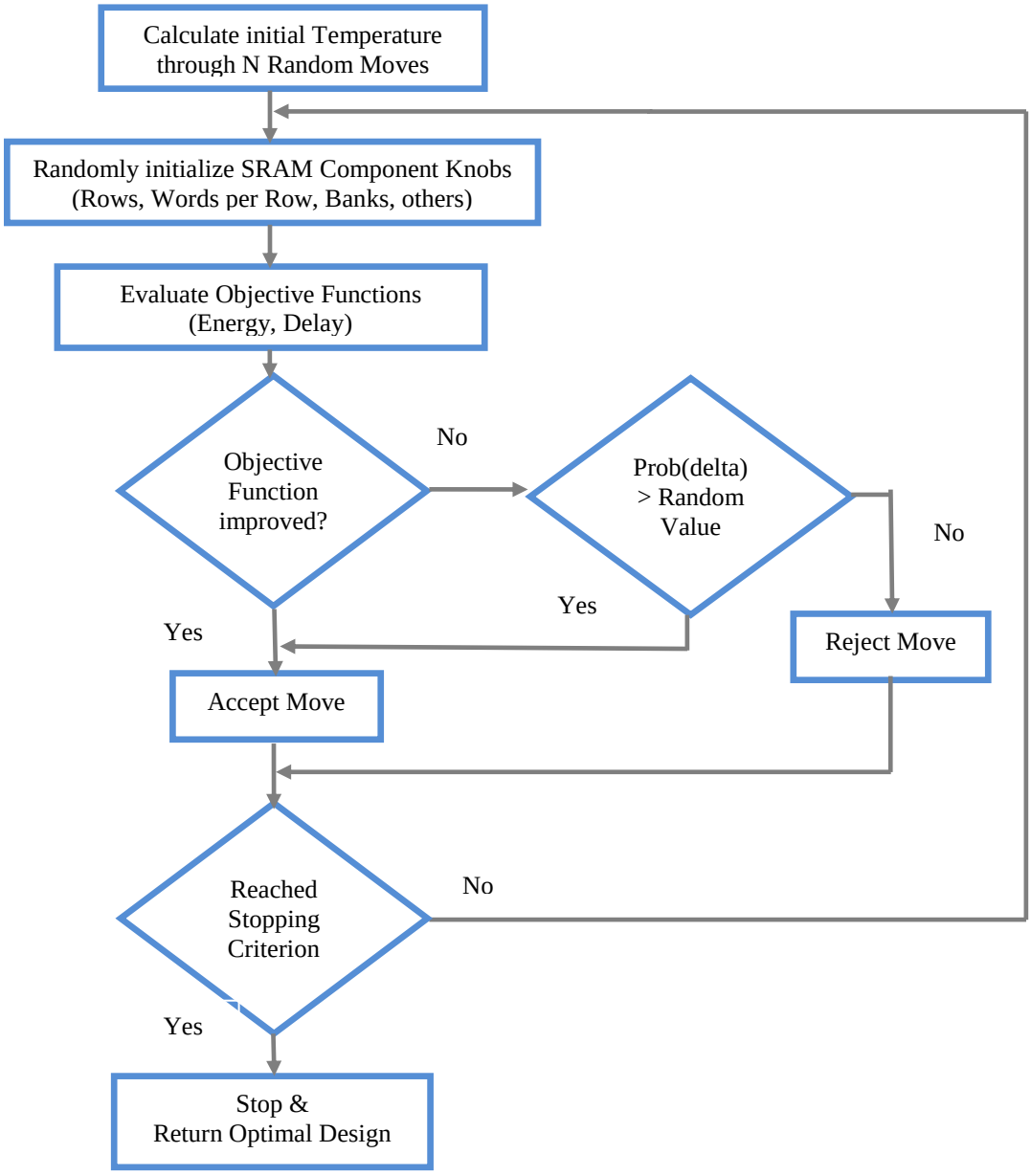


Figure 2. Simulated Annealing SRAM optimization flow chart

After each iteration, the algorithm decreases the temperature to minimize the probability of accepting designs with worse energy or delay. The algorithm will stop when it meets any of these stopping criterions: the maximum number of iterations to be performed, the maximum number of rejected designs, or the minimum temperature.

The simulated annealing algorithm is not efficient for small space optimization. We modified the algorithm as shown in Figure 3 to adapt according to the design space. If the total number of

possible combinations is less than 35, we record the data from each iteration to avoid the potential of resimulating the same design. We support flexible movement in this case to avoid being locked in one state.

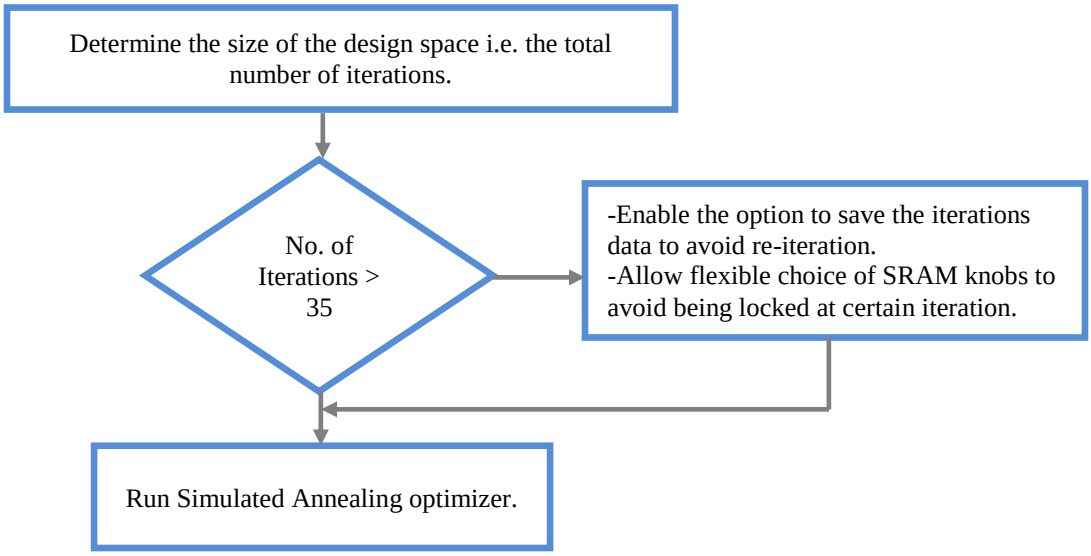


Figure 3.Adaptive simulated annealing flow chart.

The tool supports two options for optimization. The first option is to specify weights for the optimization functions (energy and delay). If weights of 1, 0 are specified for energy and delay, then the minimum energy design point will be chosen. Figure 4 shows pareto points generated from running ViPro on an actual SRAM optimization problem for demonstration. Point1 is the min energy point. If equal weights are set, then point 2 will be chosen.

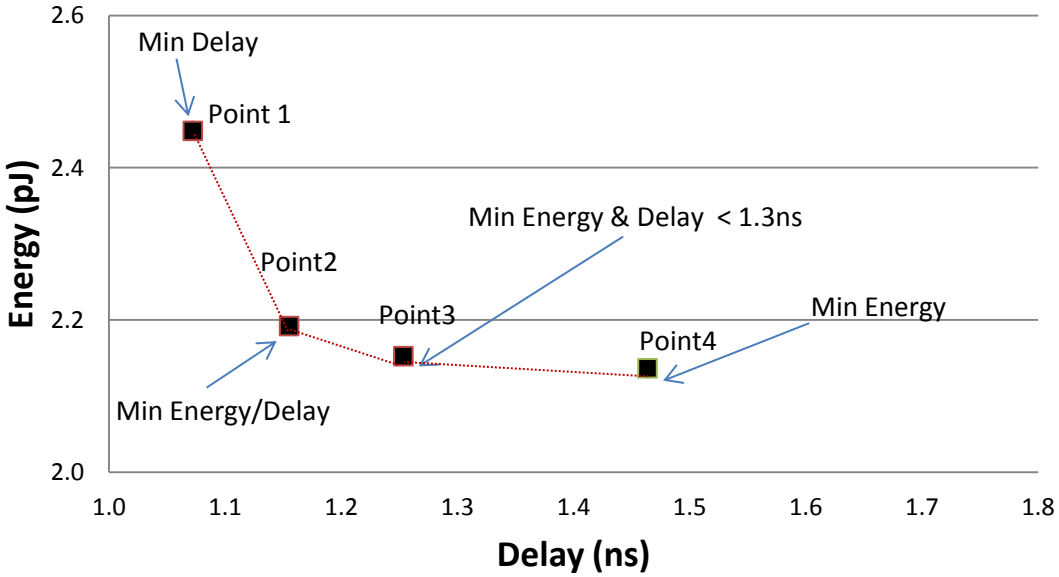


Figure 4.Different options for optimizations.

The other option is specifying a value for either the maximum energy or the delay. For example, point3 was located when a max delay of 1.3ns was specified. The optimization knobs currently supported are the architectural level parameters (number of banks, rows and words per row), two bitcells (6T and 8T), two sense amps (standard differential and offset compensated), reduced supply voltage and the option to apply word line boosting.

In the next sections we will show the results of running the optimization engine on different SRAM problems. The next section will show the results of optimizing the number of rows, banks and word per row across technologies. The following section will show the results of optimizing the aforementioned knobs together with assist features, reduced supply voltage and the option of utilizing an offset compensated sense amplifier.

System level optimization across technologies

In this section, we will optimize the system level parameters of multiple memory capacities, across multiple technologies. First we will compare the optimization results for a fixed memory size in a 45nm and 130nm commercial technology. We will then sweep the memory size to compare the optimal configurations

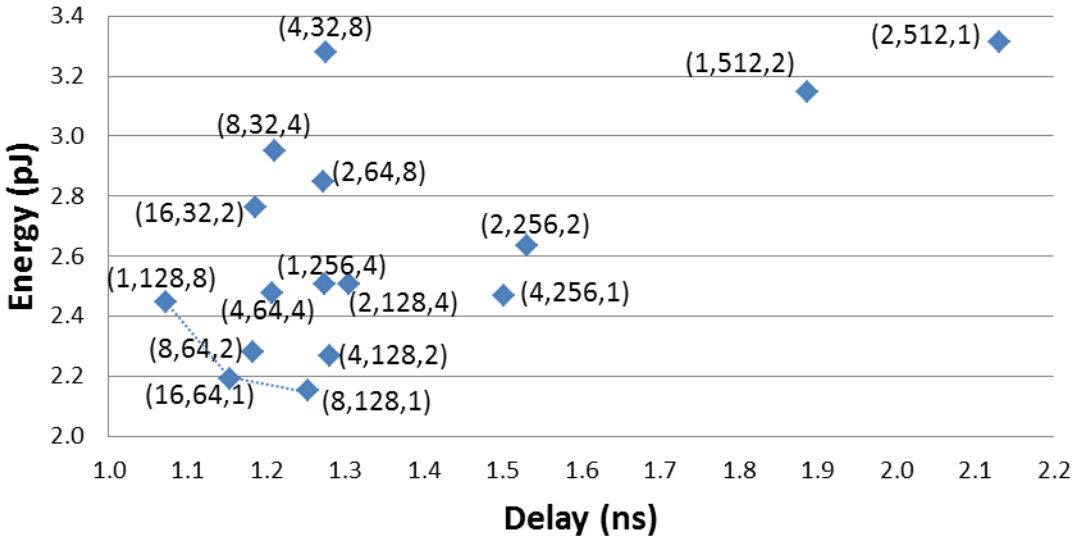


Figure 5. The pareto points for 16kB memory. Annotation format - banks, rows, words per row

The first case is to optimize a 16 Kb memory. Due to the constraints on the number of words per row, banks and rows allowed, the total number of configurations is limited to 16. Figure 5 shows the Pareto curve of the 16kB SRAM in a commercial 45 nm process. The energy is calculated as the sum of the read and write energies, and the delay is the maximum of the read and write delays. We can see from this plot that having a large number of rows increases both energy and delay. The optimal memory configuration from a delay standpoint is single bank, with 8 words per row. The minimum energy design has only one word per row, which reduces the energy wasted during half-select. We used ViPro to locate the min energy and min delay points by specifying a weight value of 1 to either variable. The average number of iterations the optimizer takes in 20 separate runs is plotted in Figure 6.

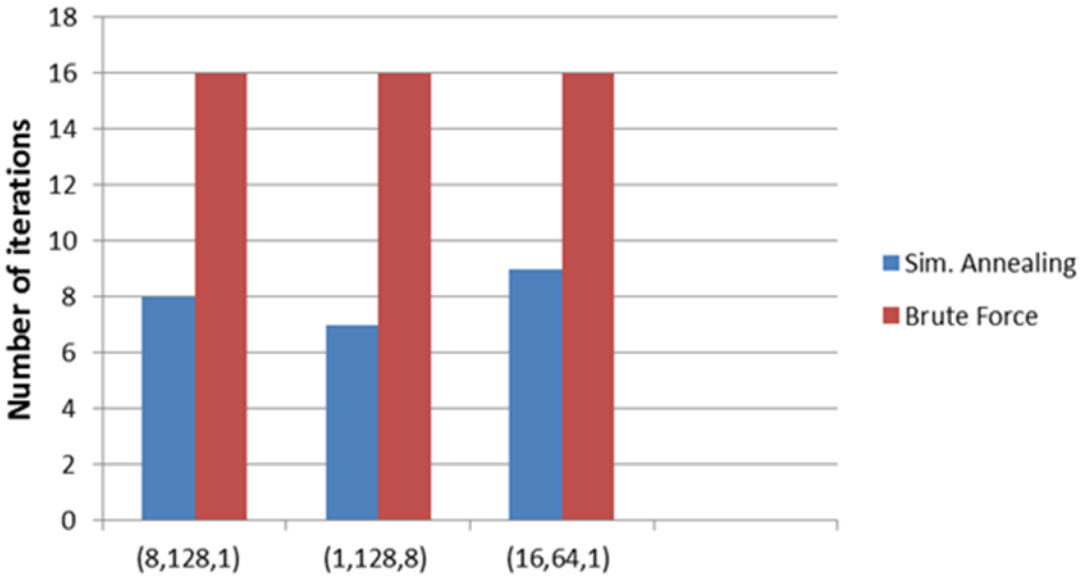


Figure 6. Comparison of the number of iterations performed by brute force and simulated annealing

This experiment was repeated for a 130 nm technology node (Figure 7). We can see from the figure that the pareto points for 45 nm are not the same in 130 nm. The 16 bank, 64 row configuration becomes the minimum energy point, while the 8 bank, 128 row configuration moves off the pareto curve. In addition, the single bank, 256 row configuration moves onto the pareto curve. This suggests…….

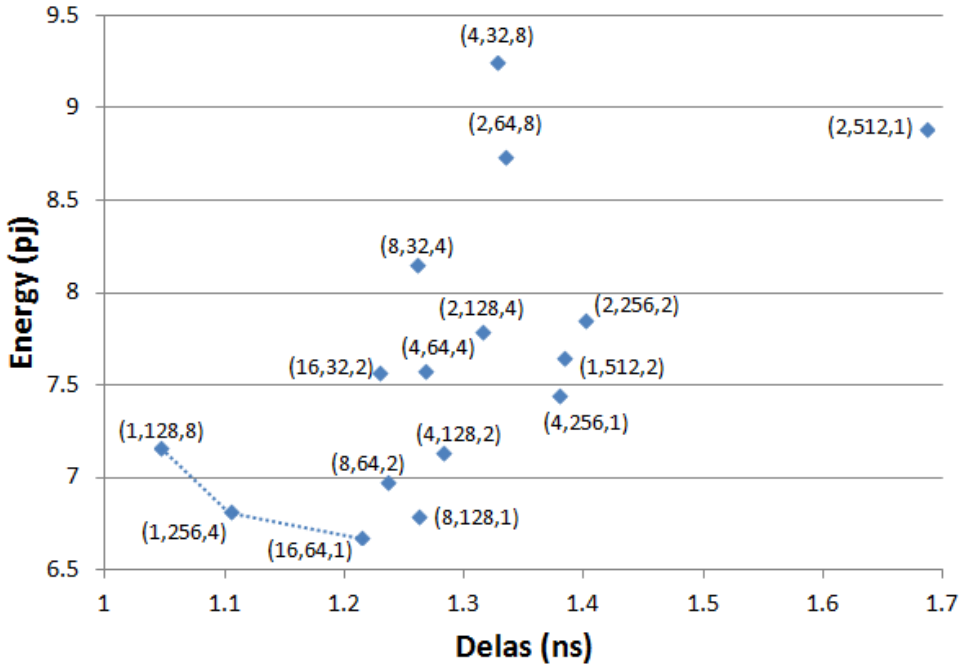
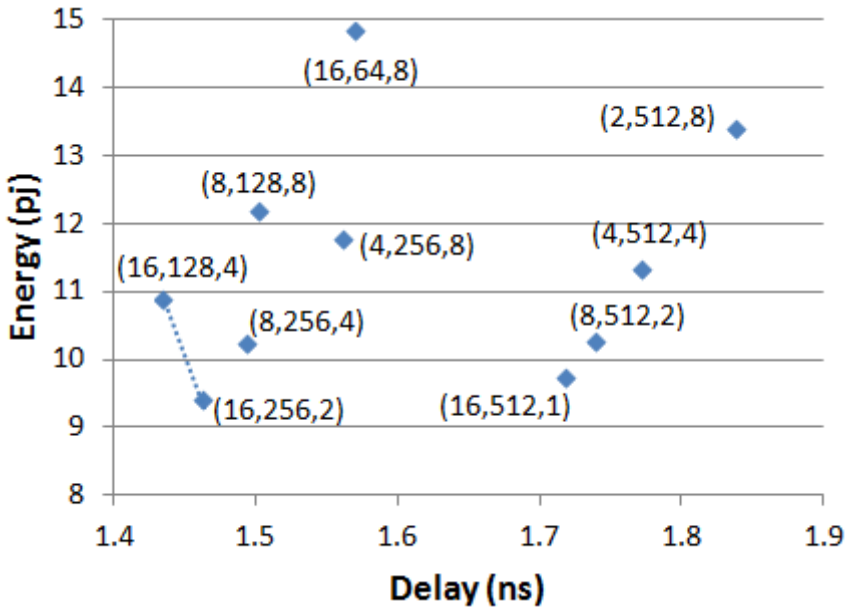


Figure 7. Optimization results in 130 nm, 16 KB memory size



In addition, we tried increasing the memory size from 16 Kb to 128 Kb. The results are shown in Figure 8. In this example, increasing the number of banks, and decreasing the number of words per row, clearly results in a lower energy/delay design.

Adding Word Line Boosting as an Optimization Knob

As SRAMs continue to scale, peripheral assist methods will be needed to allow for continued voltage scaling [4-9]. Boosting the WL above V_{DD} strengthens the passgate transistor in order to improve write-ability. The downside is that it reduces read stability in half-selected cells. However by increasing the read current through the passgate, it also reduces the read delay. The implementation of WL boosting is typically done using a charge pump circuit (Figure 9) [10]

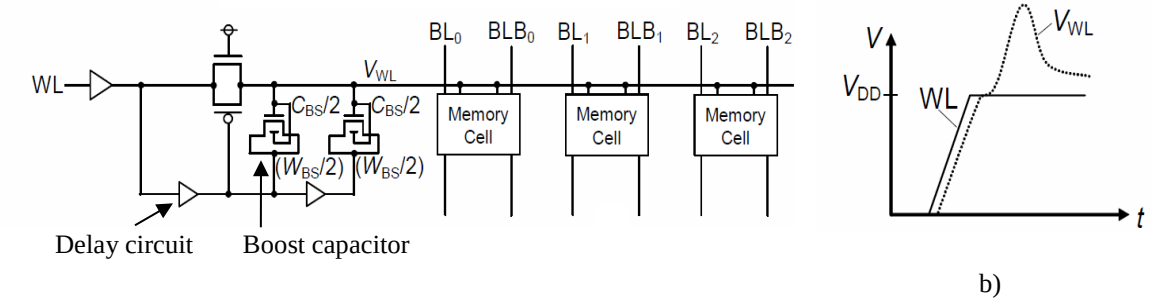


Figure 9 The word line boost circuit is shown in (a) and its operation is shown in (b).

When the WL transitions from low to high, the transmission is initially on, allowing for the output (V_{WL}) to reach full V_{DD}. Once the signal has propagated through the first delay, the capacitance of the first transistor starts to boost V_{WL} past V_{DD}. The second stage prevents V_{WL} from dropping after it has been boosted by the first capacitor [10]. The output of the word line

driver is controlled by the ratio of the boost capacitors (C_{BS}) and the capacitance of the word line as shown in equation 1:

$$V_{WL} = \left(1 + \frac{C_{BS}}{C_{WL}}\right) * V_{DD}, \quad (1)$$

In our experiments, the boost capacitors were tuned to provide 150 mV of boost to the WL voltage. This assist technique is expected to provide higher margins, allowing for V_{DD} to be reduced without sacrificing yield. The trade off is that boosting the WL above V_{DD} consumes slightly higher energy. Using ViPro, we are able to directly calculate the overhead of using a boosted WL scheme. Due to its hierarchical design, the tool can easily select between a standard WL driver and the boosted WL circuit shown in Figure 9. Once the boosted WL driver is selected, the tool automatically adjusts the WL voltage going to the bitcell to account for the boosted voltage. Currently the tool does not include yield as a global figure of merit, however this will be added in future versions.

The WL boosting circuit does not signficantly affect the optimal configurations, however it does shift the total energy slightly higher, and the worst case delay slightly lower. The average energy overhead was approximately 4.66%. The word line boosting circuit reduces the delay of the optimal configuration (1 bank, 128 rows, 8 words per row) by approximately 6.85%. While this improvement is only slight, the real advantage of using the WL boost circuit is that it decreases the critical pulse width (T_{CRIT}) required to write the bitcell [12]. This means that the margin of the worst case bitcell is greatly reduced, allowing for higher yields.

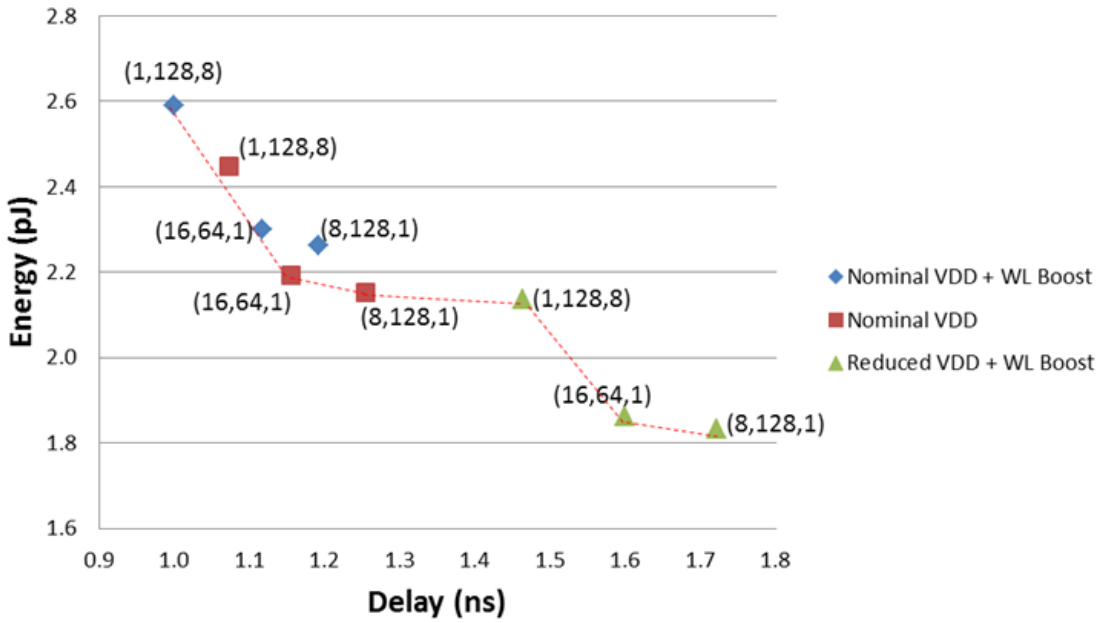


Figure 10. The pareto points for three cases are shown

Optimization Results

In this experiment, the option of reducing VDD of the memory by 100 mV is included in optimization together with the option to apply the WL boost scheme. The results in Figure 10, show the pareto optimal points for the three cases: nominal VDD with 150 mV of WL boost, nominal VDD with no WL boost, and reduced VDD with WL boost.

The interesting part of this experiment is that, if the goal is to find the minimum delay point, then the algorithm will chose to use the WL boost. However if minimum energy is the goal, then it will chose not to use the boost.

We ran ViPro to find the min energy, min delay and min energy/delay points. We also used ViPro to locate a design point with min energy with specifiying a maximum limit for the delay to be 1.13ns. The delay of the design point (16,64,1) with WL boost is 1.11ns and the delay for the same point without the WL boost is 1.15ns. The optimizer located the one with WL boosting. The number of iterations is shown in Figure 11. The number of iterations performed by brute force is 48. The average number of optimization iterations performed is 15. The optimizer results in a total speed up of ~3.5 over brute force.

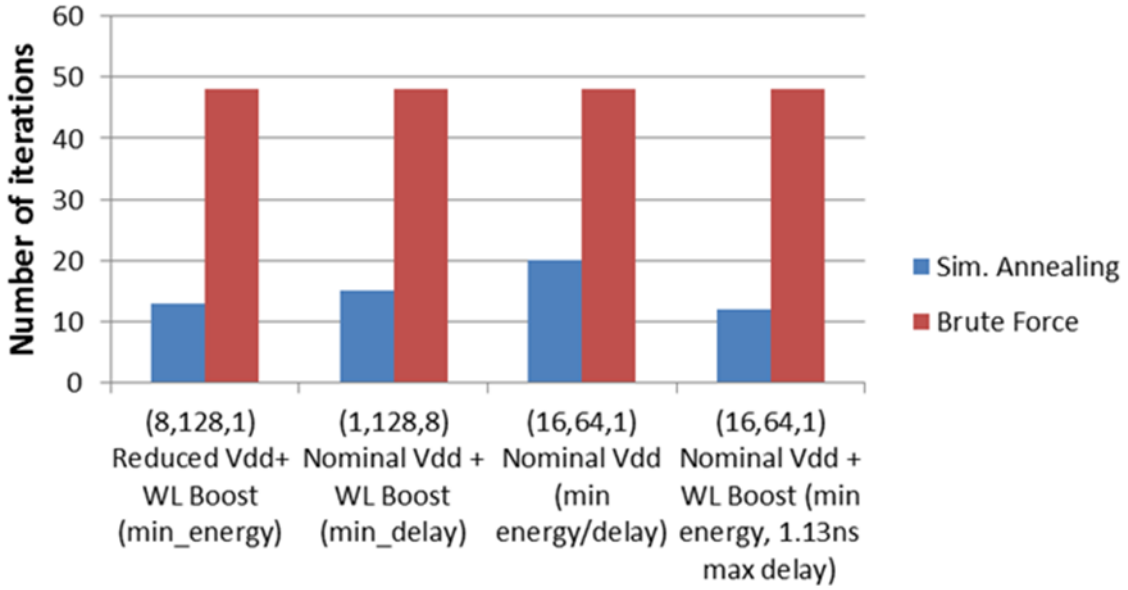


Figure 11. Comparison of the number of iterations performed by brute force and simulated annealing.

Dynamic auto-zeroing (DAZ) Offset Compensated Sense Amplifier

Voltage offset in SRAM sense amplifiers due to variability causes increased power consumption and degraded performance. In this report we will investigate the effect of utilizing the dynamic auto-zeroing (DAZ) offset compensated sense amplifier in the SRAM memory [11]. The offset compensated scheme used in this report, uses a split-phase clock and charge pump feedback circuit. Figure 12a shows a conventional latch-based sense amp. Figure 12b shows the auto-zeroing circuit attached to the sense amp. The charge pump circuit is shown in Figure 13. ENI and ENO are the input voltage differential and offset tuning phases respectively. ER1 and ER2 are reset phases.

During ER1, a zero differential input is applied to the sense amp. The ENO phase then occurs, and the SA resolves based on its intrinsic offset. The sense amp output is fed to the charge pump circuit that charges the capacitor, Cp, up or down.

During ER2, the differential input is applied to the sense amp. ENI is then pulsed, and the SA resolves based on the differential input. The duty cycle of ENR1 and ENO controls the power consumption and the offset voltage. In this report, we use a 20mV and a 50mV design, with a duty cycle of 0.5 and 0.125.

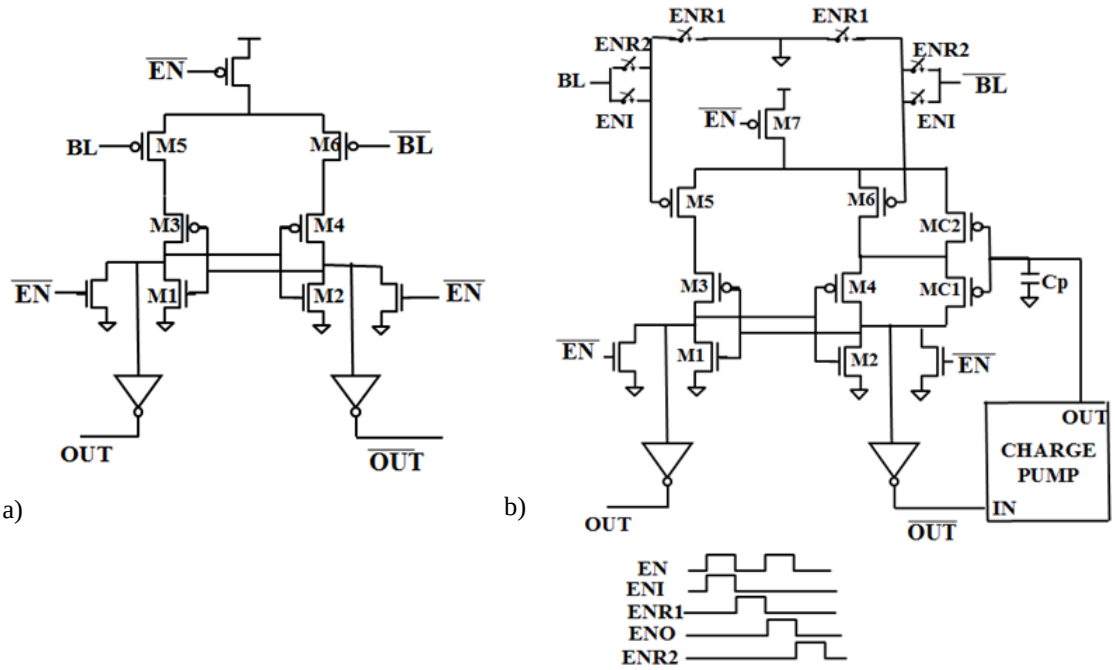


Figure 12 Schematic of the offset compensated sense amplifier

The power consumption of the offset compensated sense amp is higher than the uncompensated one due to the clock generator, charge pump and the buffer stages needed for the non-overlapping clock. The sense amplifier delay is also higher due to the high capacitive loading. On the other hand, reducing the sense amp offset reduces the necessary bit-line swing, which decreases both the precharge and bitcell energy/delay during the read operation. We anticipate that there is an optimal system design (banks, rows, words per row) that benefits from this scheme. A detailed analysis is presented in the experiments section.

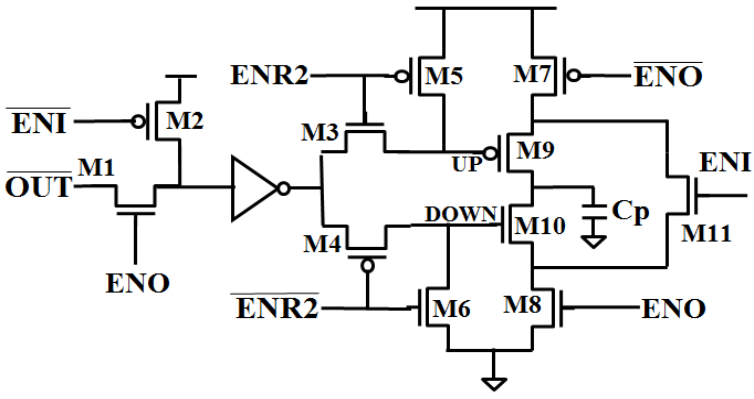


Figure 13. Charge pump circuitry used for offset compensation

Optimization Results

In this section we will first analyze the results of utilizing the offset compensated sense amplifier and then present the optimization results. Figure 14 shows the brute force results of a 16kB SRAM

using uncompensated sense amplifier and a 20mV offset compensated sense amplifier. The results indicate a big improvement in both the energy and delay for cases with large number of rows and small improvement or degradation for cases with small number of rows. The design point of 1 bank, 512 rows and 2 words per row has the biggest improvement of 10% in energy and 24% in delay. The design point of 4 banks, 32 rows and 8 words per row has the biggest degradation of 6% in energy and 5% in delay.

The offset compensated sense amplifier pushed the designed points (1, 256, 4) and (1, 128, 8) onto the Pareto curve. The improvement in energy and delay for (1, 256, 4) is ~12% and 13% respectively. For (1, 128, 8) it improves the energy by 13% and degrades the delay by 5%.

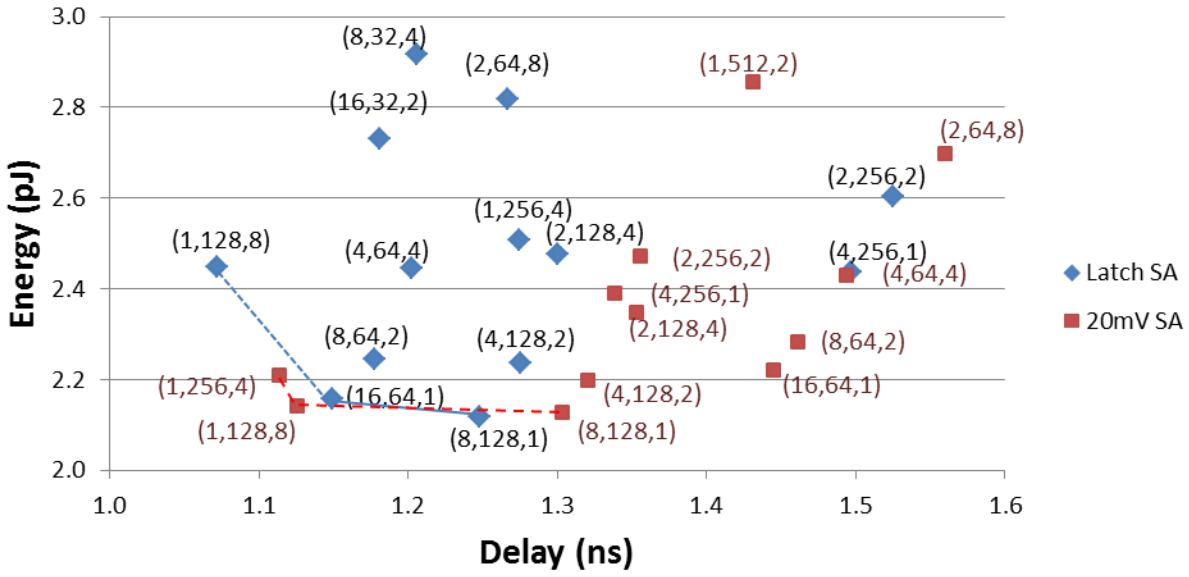


Figure 14. The Pareto curve of a 16kB SRAM memory with uncompensated and 20mV offset sense amplifier

The two Pareto optimal points show better energy and delay than any of the uncompensated sense amplifier designs. It is noticeable that the system level parameters that satisfy the energy/delay requirements changed (i.e. design point (16,64,1) is the minimum energy/delay point of the uncompensated sense amp design space). Using the offset compensated sense amplifier, the min delay/energy design changed to (1, 128, 8). Figure 15 shows the percentage reduction in the read energy saved by using the offset compensated sense amp as a function of the number of rows. The number of words per row is fixed at two in this case.

As the number of rows increases, the output capacitive load of the buffers driving the non-overlapping clocks of the sense amplifier decreases. The bitcell and precharge energy decreases because of the decrease in the bitcell swing. When the number of rows is below 128, the total read energy, in the case of using an offset compensated sense amplifier, is higher because the extra energy consumed by the non-overlapping clocks buffer chain is higher than the total reduction in energy of the bitcell and precharge cycle. When the number of rows increases above 128, the reduction of bitcell energy dominates. In addition, the energy of the clock buffers chain decreases.

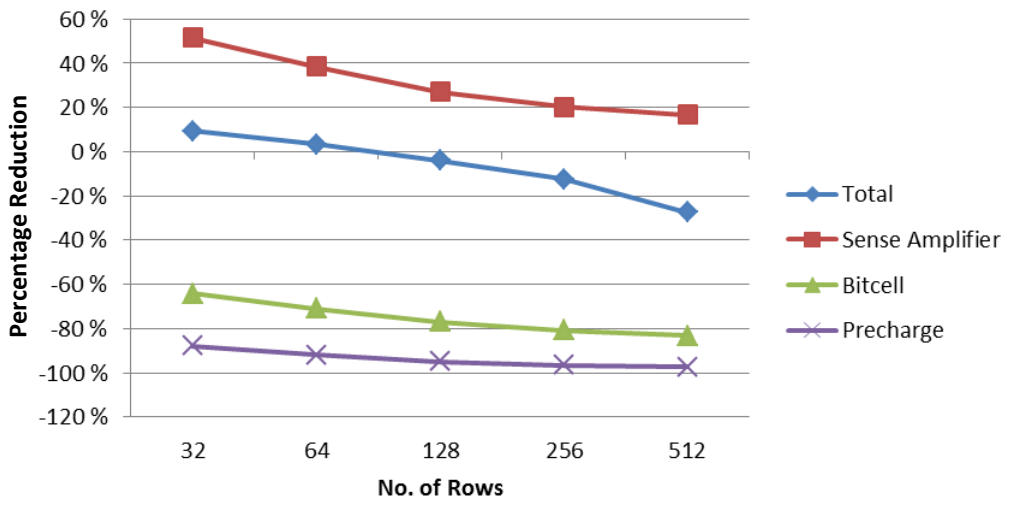


Figure 15. Percentage reduction in read energy versus the number of rows

Figure 16 shows the reduction in the read delay as the number of rows is increased. Similar to the read energy curve, the percentage reduction in read delay due to using the offset compensated sense amp is higher when the number of rows is below 128. In this case, the delay of the non-overlapping clocks buffers dominates the total delay. When the number of rows exceedes 128, the percentage reduction of delay caused by the precharge devices and the bit-cell dominates.

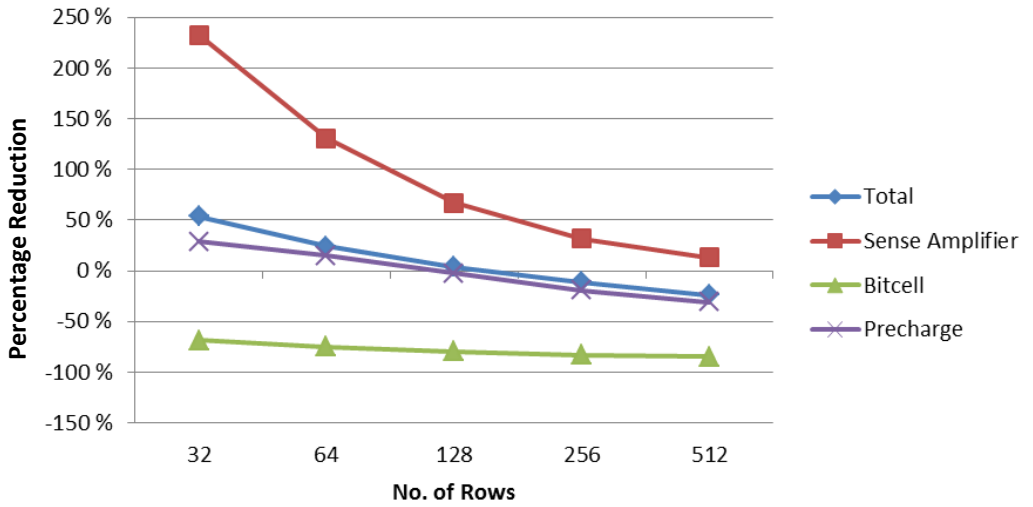


Figure 16. Percentage reduction in read delay versus the number of rows

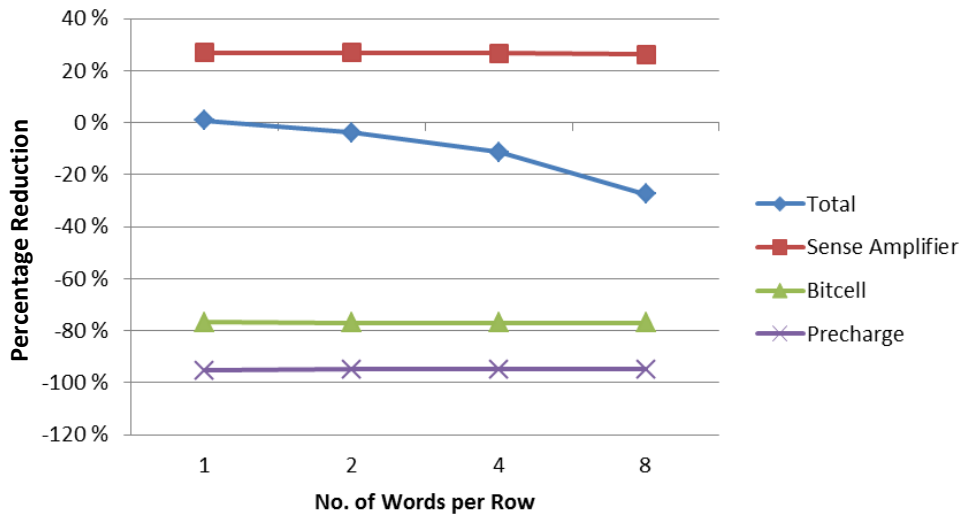


Figure 17. Percentage reduction in read energy versus the number of words per row

Figure 17 shows the percentage reduction in read energy as the number of words per row is increased. The number of rows is set constant to 128 in this case. The offset compensated sense amplifier adds a constant extra energy of ~0. 05pJ compared to the uncompensated case due to the clock generator, charge pump and clocks buffer. The percentage reduction in bitcell energy increases with the number of words per row due to the additional half-selected bitcells. Figure 18 shows the reduction in the read delay is held constant ~0.05ns, and is therefore not affected by the number of words per row.

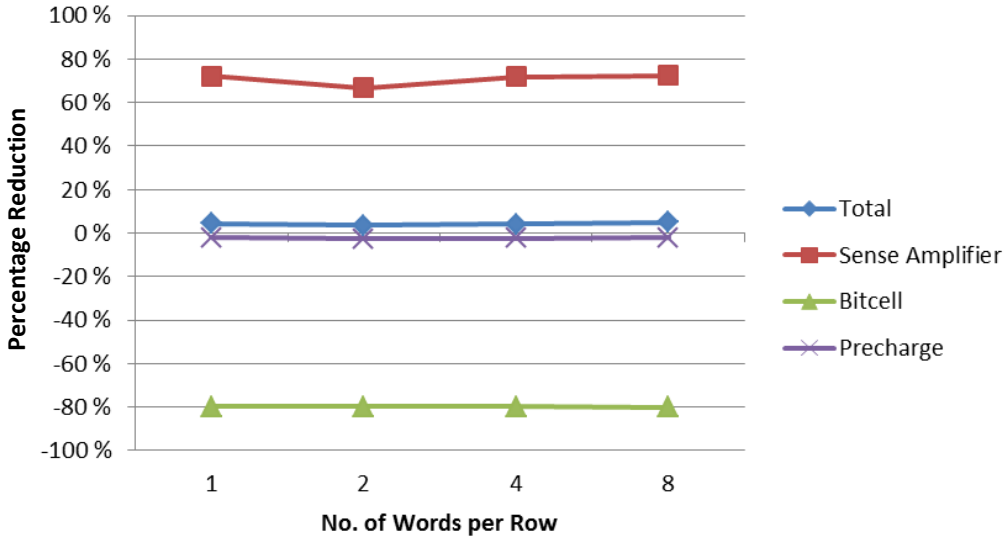


Figure 18. Percentage reduction in read delay versus the number of words per row

Figure 19 shows the brute force results of a 16kb SRAM utilizing a 50 mV versus 20 mV offset compensated sense amplifier. The 50 mV case has smaller improvements in both energy/delay for designs with high number of rows and words per row.

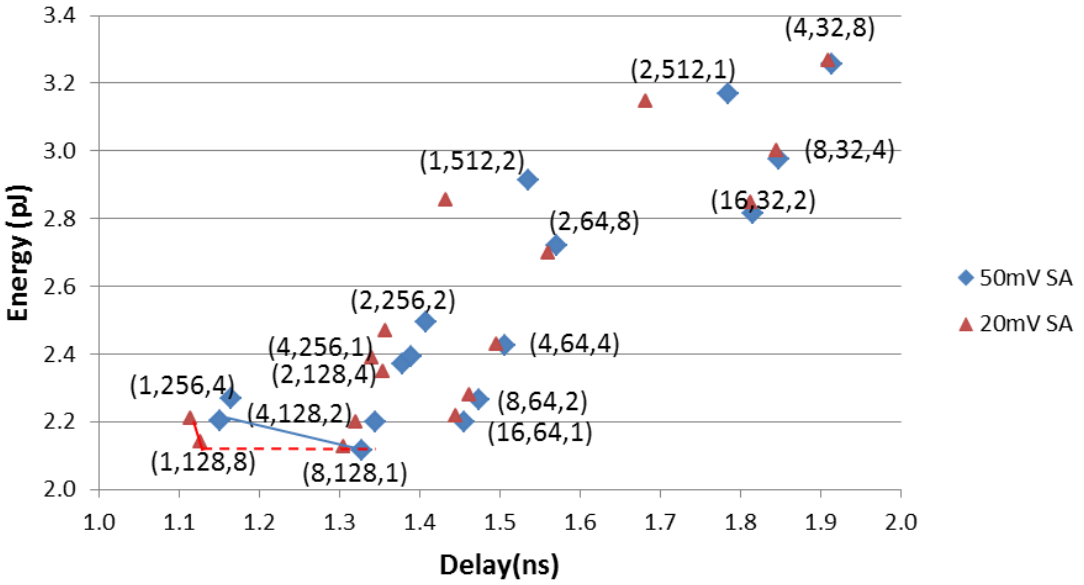


Figure 19. Pareto curve for a 16 Kb memory with 20mV and 50mV sense amplifier

We used ViPro to optimize a 16kB SRAM design including all the following options: non-compensated sense amplifier, 20 mV offset compensated sense amp, 50 mV offset compensated sense amp, and a 100 mV VDD reduction with WL boosting. The design space is shown in Figure 20.

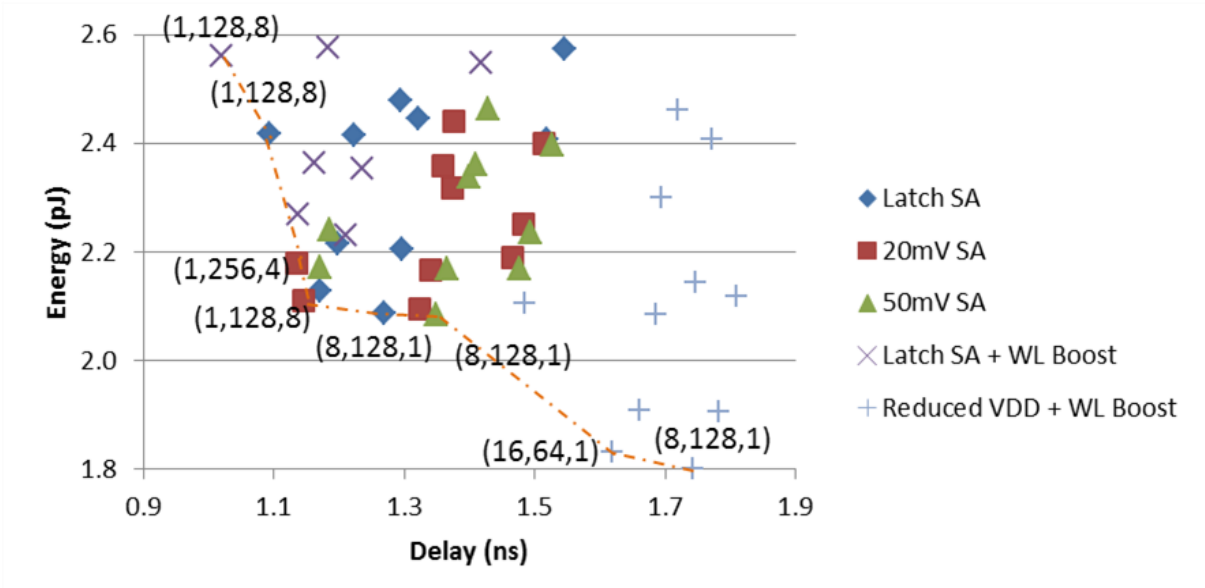


Figure 20. Pareto curve for a 16 Kb memory for five cases

We used the simulated annealing algorithm provided by ViPro to locate min energy, min delay, and min delay/energy designs. We also used it also to locate the design point with minimum energy and max delay of 1.3 ns. The same design point was located using weights of 0.4 and 0.6 for energy and delay respectively. As shown in Figure 21, the number of iterations needed using brute force was 80. The average number of iterations performed by the optimizer was 20, resulting in an average speed up of approximately 4X.

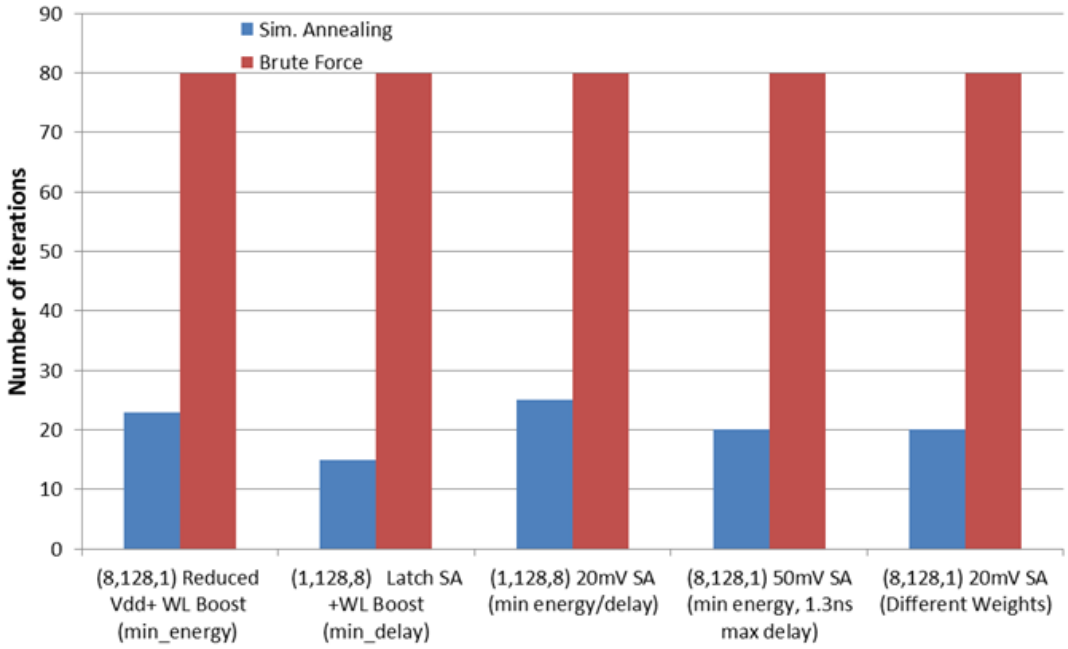


Figure 21. Comparison of the number of iterations performed by brute force and simulated annealing.

Conclusions

In conclusion, this report describes the SRAM optimization flow supported across technologies and memory sizes. We have successfully extended ViPro to optimize design spaces, with a speed up of up to 4X. It is important to note that the speed up increases as the size of the design space increases, which is a nice feature with the expected growth of the memory design space. Further, we have successfully demonstrated the effect of utilizing an offset compensated sense amplifier on the memory total energy and delay. The interesting point is that utilizing the offset compensated sense amplifier changes the optimal memory system configuration in terms of energy and delay. The effect of the word line boosting was also shown, along with the capability to rapidly locate design points on the Pareto curve that satisfy the design specifications.

References

[1] S. Nalam, M. Bhargava, K. Mai, and B. H. Calhoun, "Virtual Prototyper (ViPro): An early design space exploration and optimization tool for SRAM designers," DAC, pages 138-143, 2010.

[2] Nalam, S., M. Bhargava, K. Ringgenberg, K. Mai, and B. H. Calhoun, "A Technology-Agnostic Simulation Environment (TASE) for Iterative Custom IC Design across Processes", ICCD, pp. 523-528, 2009.

[3] Verma, N. Chandrakasan, A.P., "A 256 kb 65 nm 8T subthreshold SRAM employing sense-amplifier redundancy," IEEE J. Solid-State Circuits, pp. 141-149, 2008.

[4] K. Nii, et al., "A 45-nm single-port and dual-port SRAM family with robust read/write stabilizing circuitry under DVFS environment," IEEE Symposium on VLSI Circuits, pp. 212-213, 2008.

[5] Y.H. Chen, et al., "A 0.6V 45nm adaptive dual-rail sram compiler circuit design for lower VDD min VLSIs," IEEE Symposium on VLSI Circuits, pp. 210-211, 2008.

[6] M. Iijima, K. Seto, M. Numa, A. Tada, T. Ipposhi, "Low power sarm with boost driver generating pulsed word line voltage for sub-1V operation," Journal of Computers, pp. 34-40, 2008.

[7] N. Shibata, H. Kiya, S. Kurita, H. Okamoto, M. Tan'no, T.A. Douseki, "A 0.5V 25 MHz 1-mw 256-kb MTCMOS/SOI SRAM for solar-power-operated portable personal digital equipment – sure write operation by using step-down negatively overdrive bitline scheme," IEEE J. Solid State Circuits, pp. 728-742 2006.

[8] R.W. Mann, et al., "Impact of circuit assist methods on margin and performance in 6t sram," Solid State Electronics, pp. 1398-1407, 2010.

[9] V. Chandra, C. Pietrzyk, and R. Aitken, "On the efficacy of write-assist techniques in low voltage nanoscale srams," DATE, pp. 345-350, 2010.

[10] M. Iijima, et al., "Low power SRAM with boost driver generating pulsed word line voltage for sub-1V operation," Journal of Computers, pp. 34-40, 2008.

[11] P. Beshay., J. Ryan, and B. H. Calhoun "Sub-threshold Sense Amplifier Compensation Using Auto-zeroing Circuitry," Sub-threshold Microelectronics Conference, 2012

[12] S. Nalam, V. Chandra, R. Aitken, B. Calhoun, "Dynamic write limited minimum operating voltage for nanoscale SRAMs," DATE, pp. 1-6, 2011.